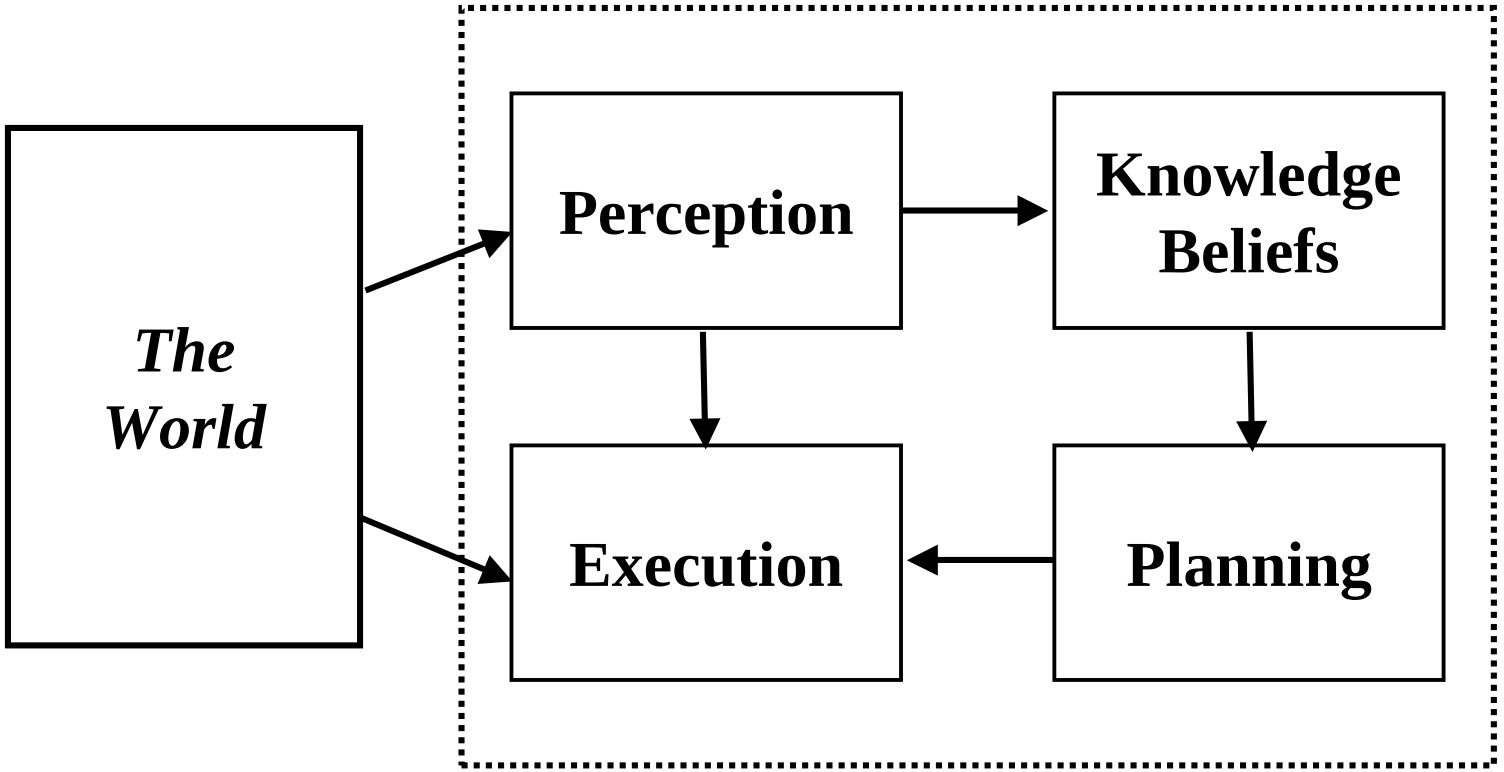
Planning, Execution & Learning
1. Introduction –
Representation and Search

Reid Simmons

Administrative Business

- Web Site: www.cs.cmu.edu/~reids/planning
- Email: reids@cs.cmu.edu; mmv@cs.cmu.edu
- Office: NSH 3205 (Reid); Wean 7123 (Manuela)
- Readings: No Textbook; Research articles posted on web site
- Evaluation:
  - 3 Homework Assignments (15% each)
  - Term Project (30%)
  - Final Take-Home Exam (25%)
- ***No Class September 19 (next Wednesday);***
  Makeup Class Monday October 22 (mid-term break)

An Agent Architecture



Why is Planning Hard?

- The Problem:
  - Find a set/sequence of actions that can transform an initial state of the world to a goal state
  - Alternately: Achieve a *set* of goals
- Why Difficult?
  - Uncertainty about environment (e.g., initial state)
  - Uncertainty about effects of actions
  - Other agents / external events can affect goal achievement
  - Agents own actions can have bad effects ("goal interactions")
  - Time and resource constraints

How to Make Planning Easier?

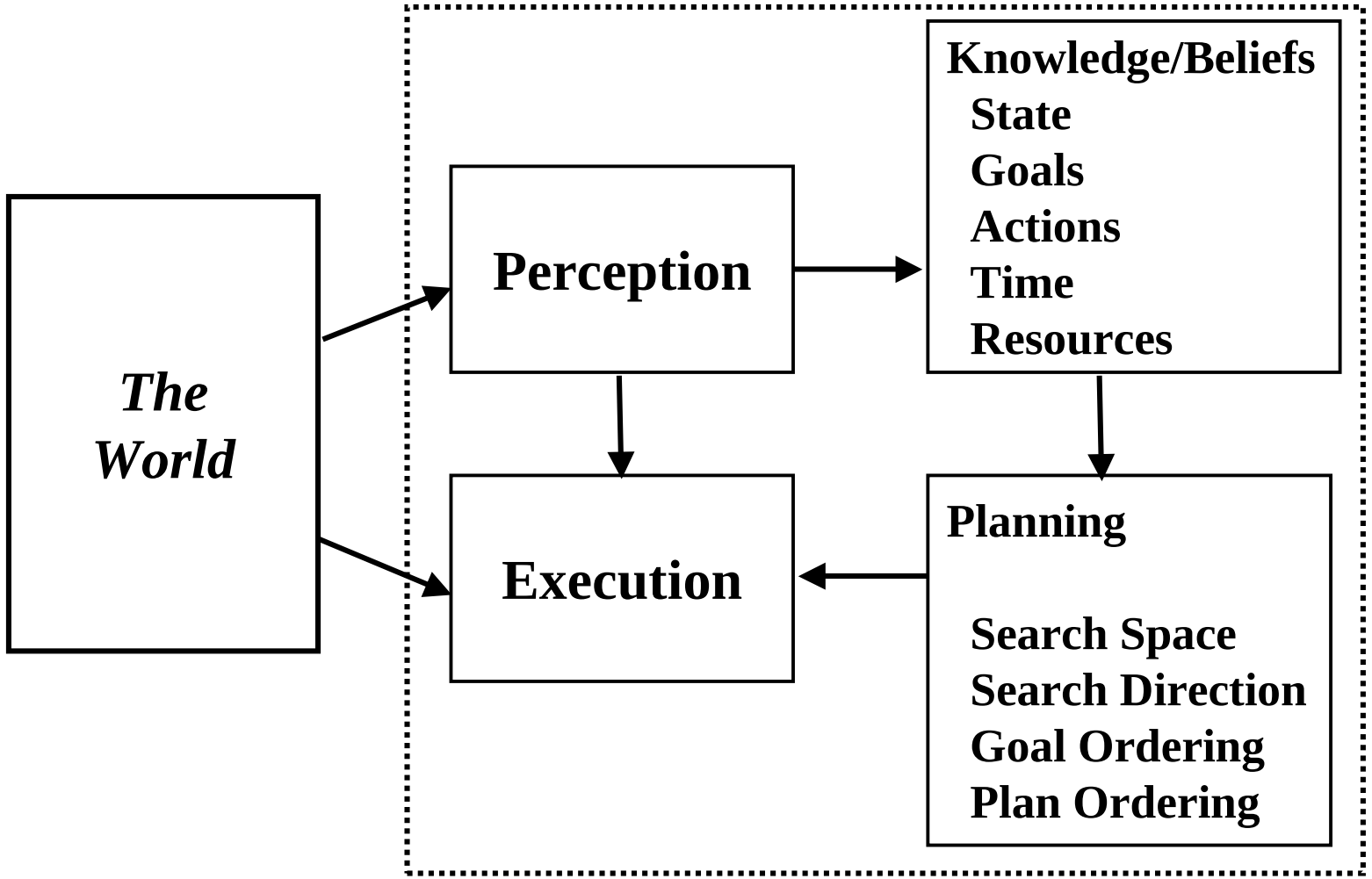
- ***Take Advantage of Characteristics of the Problem to Make Search Simpler***

- Explicit Representations of State, Goals, Actions and Plans
  - Can focus search on actions that desired given subgoals
  - Specialized algorithms can operate more efficiently
- Goal Decomposition / "Divide and Conquer"
  - Assume conjunctive goals achieved nearly independently
  - Typically, planning problems are not puzzles
- Flexible Search Strategies
  - Order in which problem is solved not necessarily order in which plan is executed

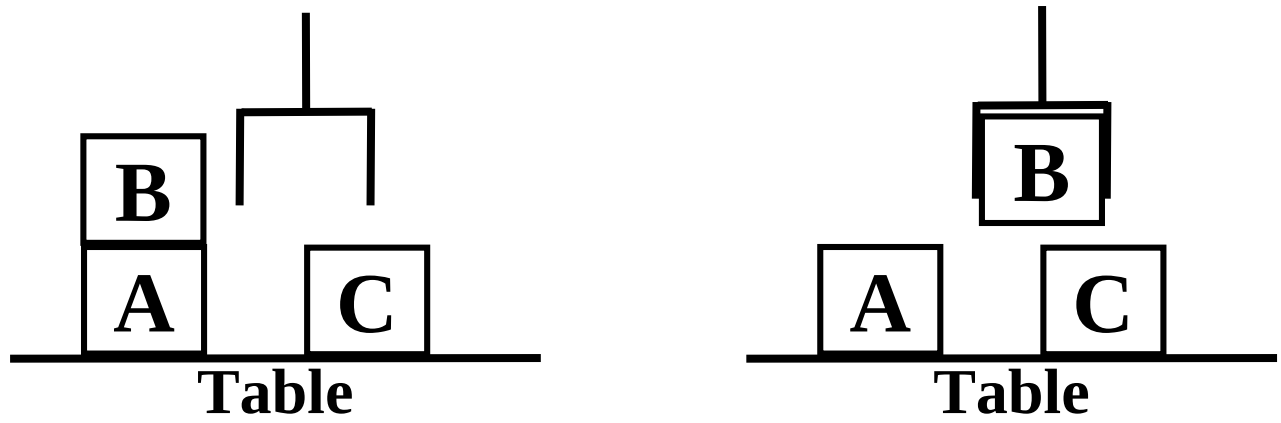
Simplifying Assumptions

- Known Initial State
- Deterministic Actions
- Single Agent / No External Events
- Simple Action Representation
  - No conditional effects
  - No quantified effects
  - No functional effects
- No Concurrent Actions
- No Sensing Actions (Implies no *branching* points in plans)
- No Deadlines and Sufficient Resources

Representation and Search



The Blocks World I



- All blocks of equal size
- Fixed table; Block position on table does not matter
- At most one block on top of another
- Any number of blocks on table
- Blocks are picked up and put down by the arm
- Arm can hold only one block at a time

The Blocks World II

- Objects
  - Blocks: *A*, *B*, *C*
  - Table: *Table*
- States
  - Conjunctions of ground literals
  - *On(A, B)*, *On(C, Table)*, *Clear(B)*, *Handempty*, *Holding(C)*
- Actions
  - Operator schemas with variables
  - *Pickup(x)*, *Putdown(x, y)*
- Domain Axioms
  - "At most one block on top of another"
  - "Hand must be empty and block must be clear to pick it up"

Logic / Situation Calculus

- Add State Variables to Each Predicate

  Clear(x, s), Handempty(s)

- Define Domain Axioms

  $\forall x, s\ Clear(x, s) \Leftrightarrow \neg\exists y\ On(y, x, s)$

  $\forall x, y, z : Block \quad On(y, x, s) \wedge On(z, x, s) \Leftrightarrow y = z$

  $\forall x: Block \quad \neg On(x, x, s)$

- **Do** Operator Maps to State Resulting from Performing Actions

  Holding(x, Do(Pickup(x), s))

- Actions are State Mappings with Preconditions and Effects

  $Handempty(s) \wedge Clear(x, s) \wedge a=Pickup(x) \Rightarrow$
  $Holding(x, Do(a, s)) \wedge (\forall y\ On(x, y, s) \Rightarrow \neg On(x, y, Do(a, s)))$

- Represent Initial and Goal States

  $On(B, A, S0) \wedge On(A, Table, S0) \wedge On(C, Table, S0) \wedge Clear(C, S0)$

  $\exists s\ On(C, A, s) \wedge On(B, Table, s)$

The Frame Problem

- Must Explicitly Declare What Does *Not* Change as a Result of Doing Actions

- Frame Axioms

$$On(x, y, s) \wedge a \neq Pickup(x) \Rightarrow On(x, y, Do(a, s))$$
$$\neg On(x, y, s) \wedge a \neq Putdown(x, y) \Rightarrow \neg On(x, y, Do(a, s))$$
$$Holding(x, s) \wedge a \neq Putdown(x, y) \wedge a \neq Drop \Rightarrow Holding(x, Do(a, s))$$
$$\neg Holding(x, s) \wedge a \neq Pickup(x) \Rightarrow \neg Holding(x, Do(a, s))$$

- Writing Frame Axioms is Tedious, Error Prone and Inefficient
  - Two axioms for each predicate
  - Length of axioms proportional to number of actions
  - No computationally tractable FOL frame axioms

STRIPS-Like Representations

- Specialized (Simplified) Representations of Actions
  - Conjunctive preconditions and effects (no conditionals)
  - No quantification
- Implicit Solution to Frame Problem
  - ***State*** is database of ground literals
  - If literal is not in database, assumed to be ***false***
  - Effects of actions represented using ***add*** and ***delete*** lists (insert and remove literals from database)
  - No explicit representation of time
  - No logical inference rules
- May Expand Branching Factor of Search Space
  - Often need to add extra actions and extra effects to compensate for simplified representation

STRIPS Meets the Blocks World

- Action Representation

Pickup_from_table(b)

Pre: Block(b), Handempty
Clear(b), On(b, Table)

Add: Holding(b)

Delete: Handempty,
On(b, Table)

Pickup_from_block(b, c)

Pre: Block(b), Handempty
Clear(b), On(b, c), Block(c)

Add: Holding(b), Clear(c)

Delete: Handempty,
On(b, c)

Putdown_on_table(b)

Pre: Block(b), Holding(b)

Add: Handempty,
On(b, Table)

Delete: Holding(b)

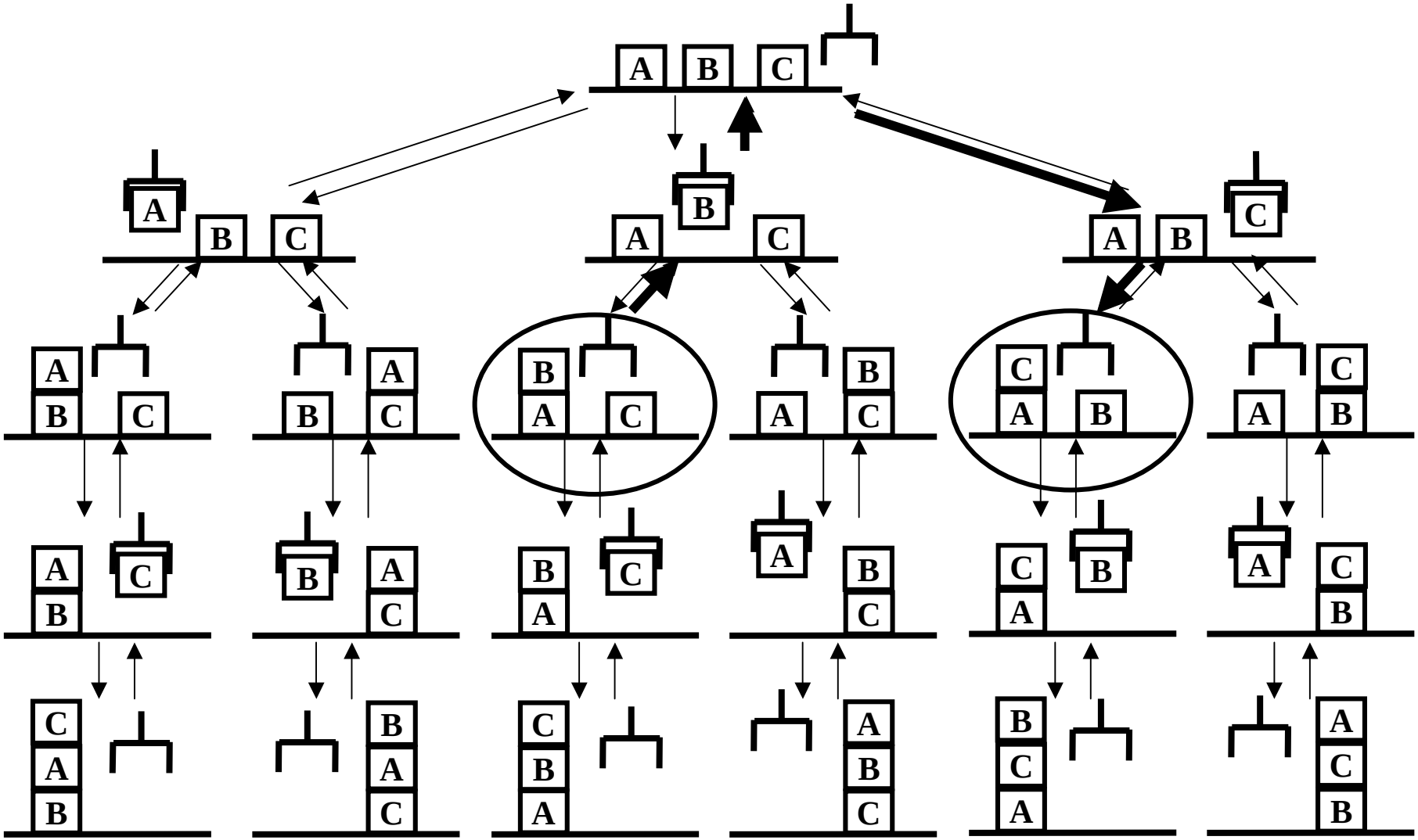
Putdown_on_block(b, c)

Pre: Block(b), Holding(b)
Block(c), Clear(c), $b \neq c$

Add: Handempty, On(b, c)

Delete: Holding(b), Clear(c)

STRIPS State Transitions



Zen and the Art of Planning

Initial: Consumed(A, Fish), Vigorous(Fish), Vigorous(Tea), Zen(A), Zen(Tea)

Goal: Vigorous(A) , Consumed(Tea, Fish)

Eat(person, thing)

Pre: Enlightened(person), Zen(thing), person $\neq$ thing

Add: Satisfied, Consumed(person, thing)

Delete: Enlightened(person), Zen(thing)

Drink(person, thing)

Pre: Zen(person), Satisfied, Consumed(person, thing)

Add: Enlightened(person), Zen(thing)

Delete: Consumed(person, thing), Satisfied

Man(person)

Pre: Zen(person), Satisfied, Vigorous(person)

Add: Enlightened(person)

Delete: Vigorous(person), Satisfied

Woman(person)

Pre: Enlightened(person)

Add: Vigorous(person), Satisfied

Delete: Enlightened(person)

More Realistic Action Representations I

- Conditional Effects

  Pickup (b)

  Pre: Block(b), Handempty, Clear(b), On(b, x)

  Add: Holding(b)
  if (Block(x)) then Clear(x)

  Delete: Handempty, On(b, x)

- Quantified Effects

  Move (o, x)

  Pre: At(o, y), At(Robot, y)

  Add: At(o, x), At(Robot, x)
  forall (Object(u)) [if (In(u, o)) then At(u, y)]

  Delete: At(o, y), At(Robot, y), forall (Object(u)) [if (In(u, o)) then At(u, y)]

- Disjunctive and Negated Preconditions

  Or[Holding(x), Not[Lighter_Than_Air(x)]]

- *All these extensions can be emulated by adding actions*

More Realistic Action Representations II

- *These extensions make the planning problem significantly harder*
- Inference Operators / Axioms

  Clear(x) iff forall(Block(y))[Not[On(y, x)]]

- Functional Effects

  Move (o, x)

  Pre: At(o, y), At(Robot, y), Fuel(f), f ≥ Fuel_Needed(y, x)

  Add: At(o, x), At(robot, x), Fuel(f – Fuel_Needed(y, x)),
  forall (Object(u)) [if (In(u, o)) then At(u, y)]

  Delete: At(o, y), At(Robot, y), Fuel(f),
  forall (Object(u)) [if (In(u, o)) then At(u, y)]

More Realistic Action Representations III

- *These extensions make the problem even harder still*
- Disjunctive Effects

  Pickup_from_block(b)

  Pre: Block(b), Handempty, Clear(b), On(b, c), Block(c)

  C1: Add: Clear(c), Holding(b); Delete: On(b, c), Handempty

  C2: Add: Clear(c), On(b, Table); Delete: On(b, c)

  C3: Add: ; Delete:
- Probabilistic Effects
  - Add probabilities to contexts of disjunctive effects
- Other Extensions
  - External events — Sensing actions
  - Concurrent events — Actions with duration

Search Techniques for Planning

- Planning Involves ***Search*** Through a ***Search Space***
  - How to conduct the search
  - How to represent the search space
  - How to evaluate the solutions

- Non-Deterministic ***Choice Points*** Determine Backtracking
  - Choice of actions
  - Choice of variable bindings
  - Choice of temporal orderings
  - Choice of subgoals to work on

Progression vs. Regression

- **Progression (forward-chaining):**
  - (Non-deterministically) choose action whose preconditions are satisfied
  - Continue until goal state is reached
- **Regression (backward-chaining):**
  - (Non-deterministically) choose action that has an effect that matches an unachieved subgoal
  - Add unachieved preconditions to set of subgoals
  - Continue until set of unachieved subgoals is empty

Progression: + Simple algorithm ("forward simulation")
- Often large branching factor
- Unfocused search

Regression: + Focused on achieving goals
+ Often more efficient
- Need to reason about actions
- Regression is incomplete, in general

Linear vs. Non-Linear

- **Linear:**
  - Solve one goal at a time
  - Search with a ***stack*** of unachieved goals
- **Non-Linear:**
  - Interleave attending to subgoals
  - Search with a ***set*** of unachieved goals

Linear:
+ Simple search strategy
+ Efficient ***if*** goals are indeed independent
- May produce suboptimal plans
- ***Incomplete***

Non-Linear:
+ Complete
+ Can produce shorter plans
- Larger search space
(does not take advantage of goal independence)

State-Space vs. Plan-Space

- **State-Space:**
  - Search space is a set of states of the world
  - Transitions between states are ***actions***
  - Plan is a path through the space
- **Plan-Space:**
  - Search space is a set of ***plans*** (including partial plans)
  - Initial state is ***null plan***
  - Transitions are ***plan operators*** (ad action, add ordering, etc.)

State-Space: + Easy to determine which subgoals are achieved and which actions are applicable
- Intractable to represent concurrent actions

Plan-Space: + Search order not same as plan execution order
- Hard to determine what is true in a plan

Advantages to maintaining both *state* and *plans* (ala Prodigy)

Total vs. Partial Order

- **Total Order:**
  - Plan is always a strict sequence of actions
- **Partial Order:**
  - Plan steps *may* be unordered
  - Plan may be ***linearized*** prior to execution

Total Order: + Simpler planning algorithm
- No concurrent plans
- May be forced to make unnecessary decisions

Partial Order: + Least commitment
+ Easily handles concurrent plans
- Hard to determine which goals are achieved at any given time
- More complex set of plan operators